Life Cycle Management (LCM)

Proposed Technical Specification

Version 1.0.A

Revision History

Version	Date	Author	Description
1.0.A	2018.12.27	Vodafone Group Services GmbH	Minor corrections to the following diagrams: Use-Cases, Architecture, EE Constructs and State Machine.
1.0	2018.12.07	Vodafone Group Services GmbH	First official LCM specification draft release.

Content

1 Introduction

1.1 Goals

This documents aims to describe how carriers can enable their already existing set of remote managed home-gateways to support the Software Life Cycle Management (LCM) of services residing within their Home-Gateways, whilst maintaining the ability to:

1) **Dynamically launch and manage new services** without having to replace the existing firmware or breaking core functionalities.
2) **Run services in isolated and constrained environments**, without coming across security or stability issues.
3) **Take advantage of standardized APIs** and modular architectures, which promote reusability and ease of integration across different software stacks.
4) **Trigger these operations both remotely and locally**.

1.2 Target Audience

This document is a technical specification created to support carriers, ODMs, developers, testers, integrators, system engineers, project managers and product managers, to get an overall understanding of the solution.

1.3 Conventions

This document adheres to a specific taxonomy used to highlight the criticality of the different requirements.

- **MUST,** this word, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- **MUST NOT,** this phrase means that the definition is an absolute prohibition of the specification.
- **SHOULD,** this word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications must be understood and carefully weighted before choosing a different course.
- **MAY**, this word, or the adjective "OPTIONAL", means that this item is one of an allowed set of alternatives. An implementation that does not include this option MUST be prepared to inter-operate with another implementation that does include the option.

1.4 Glossary

CPE, Customer premise equipment. The modem router in this document.

SDP, Service Delivery Platform.

LCM, Life cycle manager.

EE, Execution Environment.

IPC, Inter Process Communication.

ODM, Original Device Manufacturer.

DU, Deployment Unit.

LXC, Linux Containers.

UCI, Unified Configuration Interface.

UBUS, OpenWRT micro bus architecture.

CWMP, CPE Wan Management Protocol.

LCMd, LCM daemon.

CWMPd, CWMP daemon.

1.5 References

Document	Description
prpl HL-API (LCM).xlsx	prplFoundation LCMd HL-API specification.
info_request.json	Execution Environment info request method API specification.
info_response.json	Execution Environment info response method API specification.
install_request.json	Execution Environment install request method API specification.
install_response.json	Execution Environment install response method API specification.
start_request.json	Execution Environment start request method API specification.
start_response.json	Execution Environment start response method API specification.
stop_request.json	Execution Environment stop request method API specification.
stop_response.json	Execution Environment stop response method API specification.
uinstall_request.json	Execution Environment uninstall request method API specification.
uinstall_response.json	Execution Environment uninstall response method API specification.
TR-157_Amendment-3.pdf	Remote Software Module Management via CWMP/TR-069.
tr-157-1-3-0.html	TR-157 Software Modules Data Model specification.

2 LCM

2.1 Features

Software Life Cycle Management frameworks should enable carriers to perform the typical create, read, update and delete (CRUD) based operations on services, which are typically encapsulated in packages as depicted on the picture bellow.

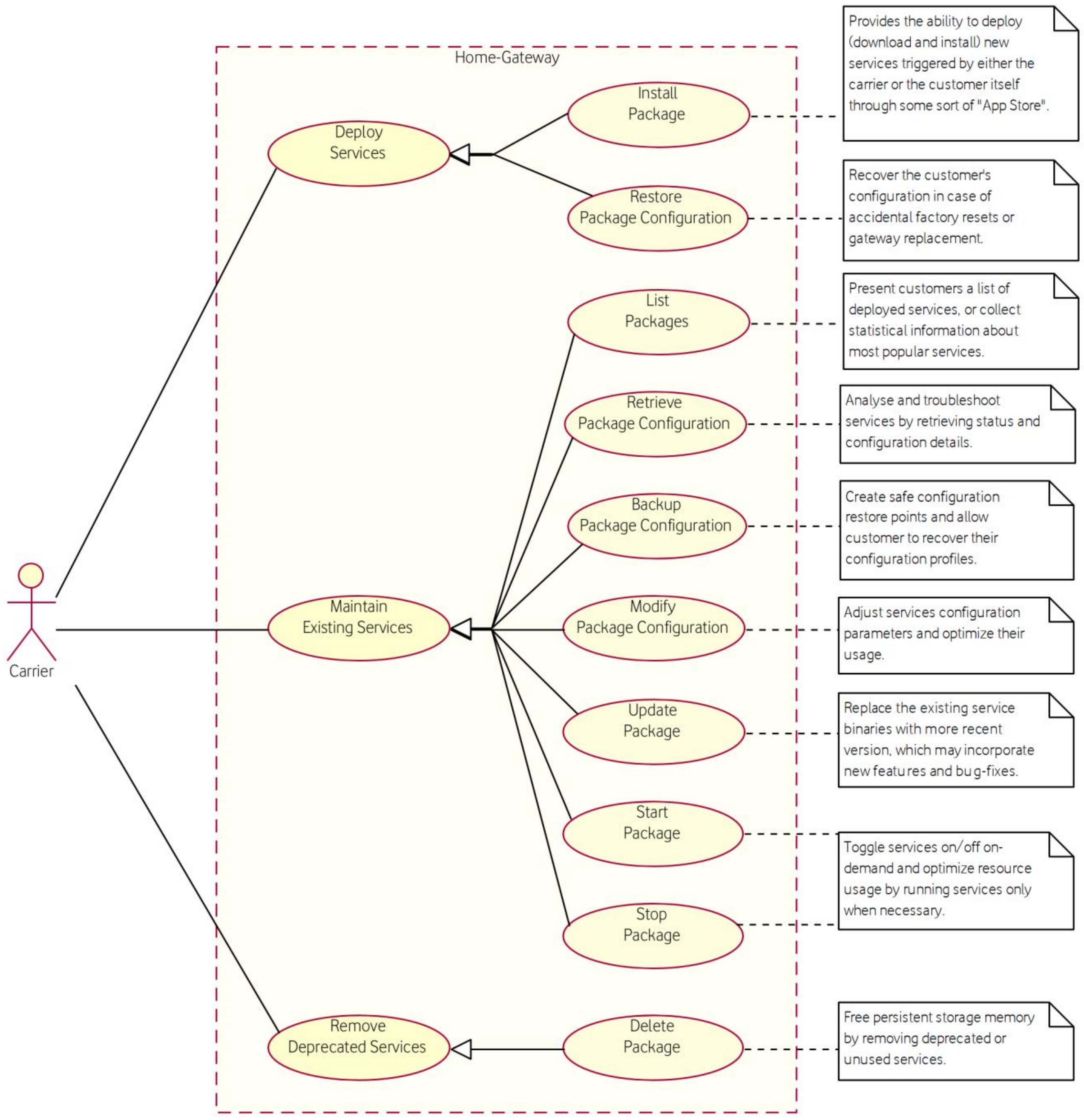

Home-Gateway
Carrier
Deploy Services
Maintain Existing Services
Remove Deprecated Services
Install Package
Restore Package Configuration
List Packages
Retrieve Package Configuration
Backup Package Configuration
Modify Package Configuration
Update Package
Start Package
Stop Package
Delete Package
Provides the ability to deploy (download and install) new services triggered by either the carrier or the customer itself through some sort of "App Store".
Recover the customer's configuration in case of accidental factory resets or gateway replacement.
Present customers a list of deployed services, or collect statistical information about most popular services.
Analyse and troubleshoot services by retrieving status and configuration details.
Create safe configuration restore points and allow customer to recover their configuration profiles.
Adjust services configuration parameters and optimize their usage.
Replace the existing service binaries with more recent version, which may incorporate new features and bug-fixes.
Toggle services on/off on-demand and optimize resource usage by running services only when necessary.
Free persistent storage memory by removing deprecated or unused services.

Figure 1. LCM Features & Use-Cases Diagram.

2.2 Building Blocks

When providing the ability to dynamically modify the OS internal building blocks, it also becomes important to enable mechanisms, which protect the system from the typical stability issues (e.g.: system reboots, unresponsive services), and security breaches (e.g.: provide untrusted services, access to protected data).

Taking that into consideration, in addition to packages, LCM also introduces the concept of Execution Environments (EEs), which isolate and constrain the running context by enforcing file access control, and resource capping as depicted on the picture bellow.

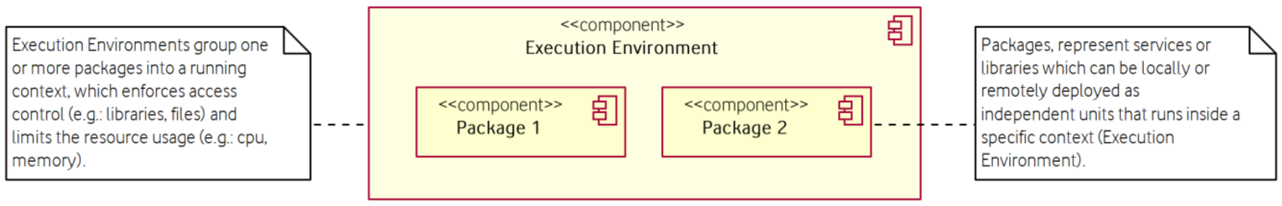

Execution Environments group one or more packages into a running context, which enforces access control (e.g.: libraries, files) and limits the resource usage (e.g.: cpu, memory).
<<component>>
Execution Environment
<<component>>
Package 1
<<component>>
Package 2
Packages, represent services or libraries which can be locally or remotely deployed as independent units that runs inside a specific context (Execution Environment).

Figure 2. LCM Constructs.

Meanwhile this mechanism grants carriers the ability to come up with custom Execution Environments, in aid of enabling packages to also be deployed directly on the root file system, LCMd must expose the default "System" EE.

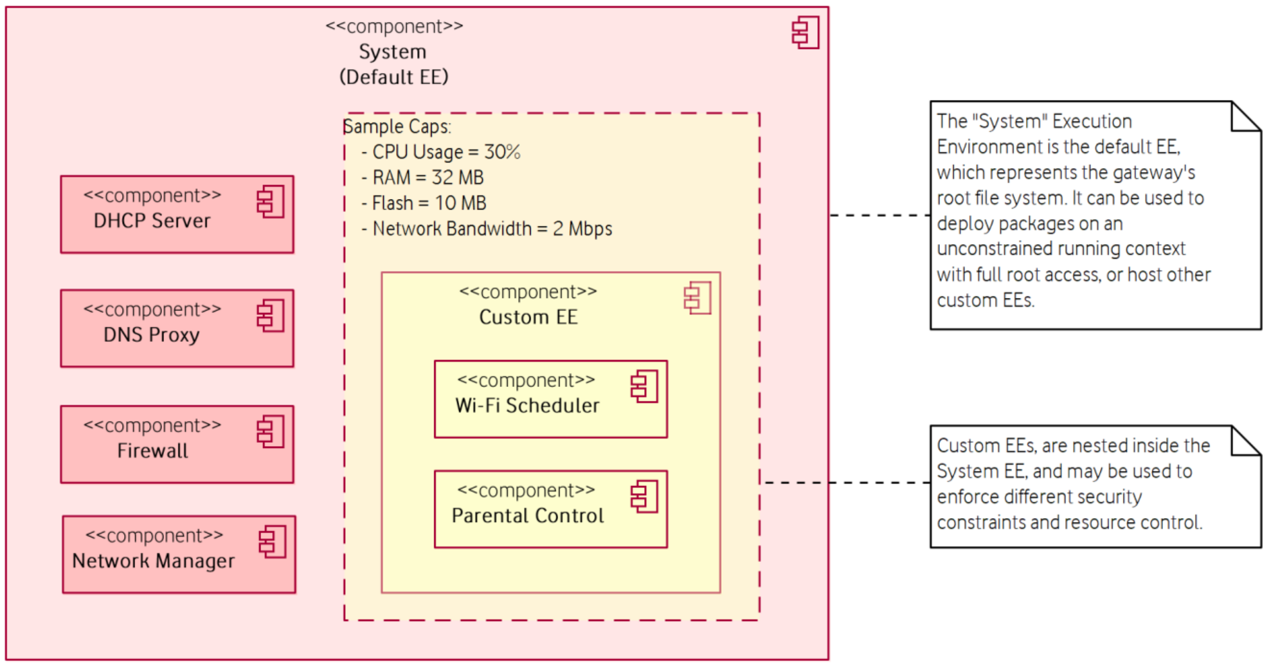
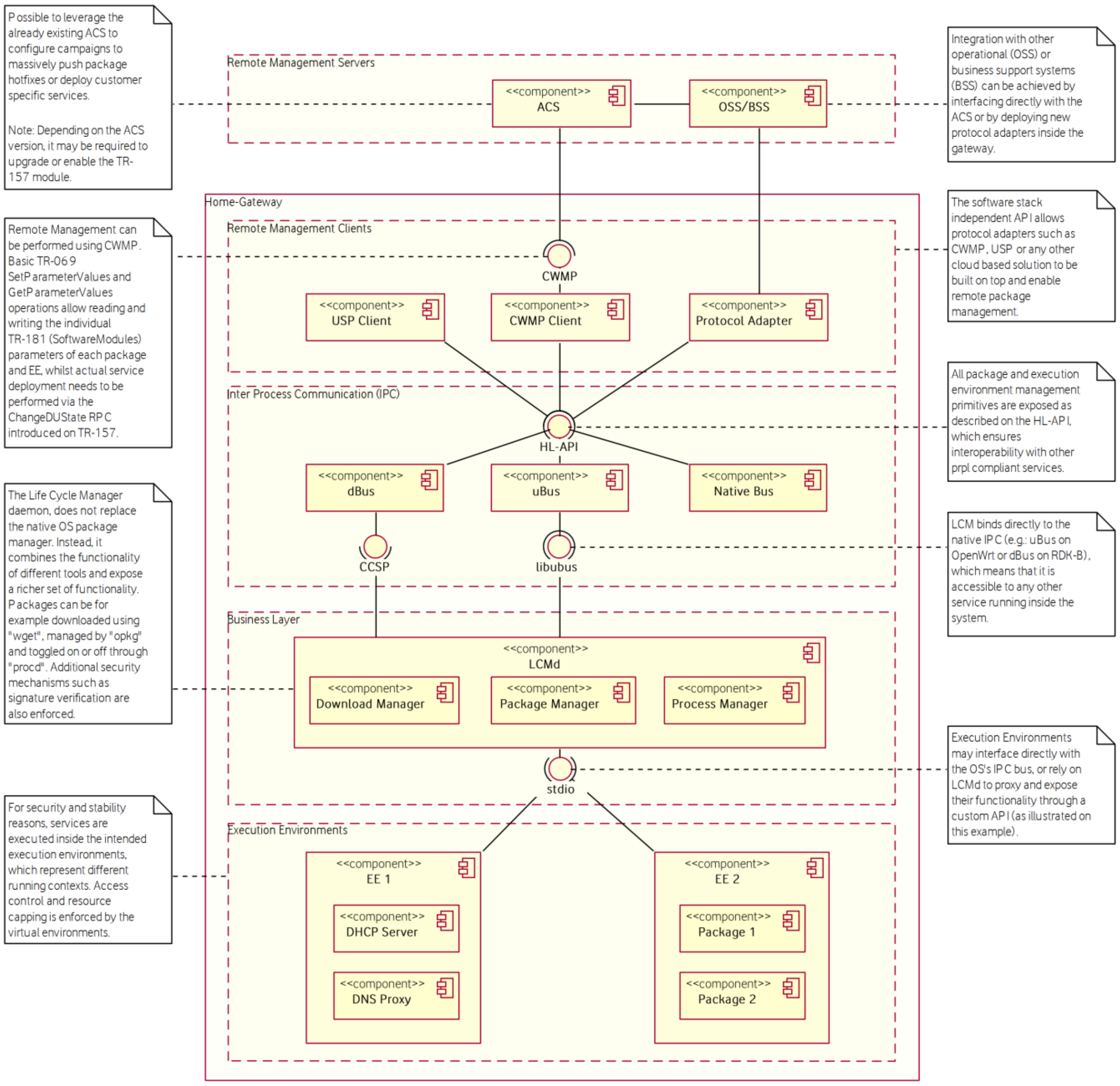

<<component>>
System
(Default EE)
Sample Caps:
- CPU Usage = 30%
- RAM = 32 MB
- Flash = 10 MB
- Network Bandwidth = 2 Mbps
<<component>>
DHCP Server
<<component>>
DNS Proxy
<<component>>
Firewall
<<component>>
Network Manager
<<component>>
Custom EE
<<component>>
Wi-Fi Scheduler
<<component>>
Parental Control
The "System" Execution Environment is the default EE, which represents the gateway's root file system. It can be used to deploy packages on an unconstrained running context with full root access, or host other custom EEs.
Custom EEs, are nested inside the System EE, and may be used to enforce different security constraints and resource control.

Figure 3. LCM Execution Environments Hierarchy.

2.1 Architecture

LCM relies on a standardized set of APIs, which enables it to be implemented across different Software Stacks, such as OpenWrt or RDK-B. Its layered and modular architecture, also promotes an ease of integration with different services and remote management protocols (e.g.: CWMP/TR-069 and USP).



Possible to leverage the already existing ACS to configure campaigns to massively push package hotfixes or deploy customer specific services.

Note: Depending on the ACS version, it may be required to upgrade or enable the TR-157 module.

Remote Management can be performed using CWMP. Basic TR-069 SetParameterValues and GetParameterValues operations allow reading and writing the individual TR-181 (SoftwareModules) parameters of each package and EE, whilst actual service deployment needs to be performed via the ChangeDUState RPC introduced on TR-157.

The Life Cycle Manager daemon, does not replace the native OS package manager. Instead, it combines the functionality of different tools and expose a richer set of functionality. Packages can be for example downloaded using "wget", managed by "opkg" and toggled on or off through "procd". Additional security mechanisms such as signature verification are also enforced.

For security and stability reasons, services are executed inside the intended execution environments, which represent different running contexts. Access control and resource capping is enforced by the virtual environments.

Integration with other operational (OSS) or business support systems (BSS) can be achieved by interfacing directly with the ACS or by deploying new protocol adapters inside the gateway.

The software stack independent API allows protocol adapters such as CWMP, USP or any other cloud based solution to be built on top and enable remote package management.

All package and execution environment management primitives are exposed as described on the HL-API, which ensures interoperability with other prpl compliant services.

LCM binds directly to the native IPC (e.g.: uBus on OpenWrt or dBus on RDK-B), which means that it is accessible to any other service running inside the system.

Execution Environments may interface directly with the OS's IPC bus, or rely on LCMd to proxy and expose their functionality through a custom API (as illustrated on this example).

Figure 4. LCM Architecture.

2.2 Remote Management

As depicted on the architecture diagram, the remote managers are decoupled from the actual Life Cycle Manager (LCM) service, which means that any remote management protocol can be built on top.

2.2.1 CPE WAN Management Protocol (CWMP)

Considering the worldwide adoption of CWMP, this chapter describes how most operations can be triggered from a TR-069 based ACS.

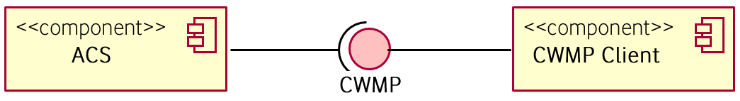


Figure 5. Remote Management Interface.

In aid of this, the following subset of Remote Procedure Calls (RPCs) needs to be supported.

Table 1. CWMP RPC Dependencies.

Depency	RPC	Description
TR-069	cwmp:GetParameterValues	Reading package and execution environment parameters (e.g.: operational state, number of deployed packages).
TR-069	cwmp:SetParameterValues	Modifying package and execution environment configuration, as well as toggling on/off services.
TR-157	cwmp:ChangeDUState	Install, update and remove packages or execution environments.

In addition to this, the following subset of TR-157 parameters needs to be supported by the CWMP Client running on the CPE.

Table 2. TR-157 Parameter Dependencies.

Parameters	Description
SoftwareModules.*	Groups statistics details in regards to the number of package and execution environments.
SoftwareModules.ExecEnv.{i}.*	Execution Environment specific parameters (e.g.: administrative state, name, allocated resources).
SoftwareModules.DeploymentUnit.{i}.*	Static package specific parameters, applicable to both service or execution environments (e.g.: UUID, Name, Vendor).
SoftwareModules.ExecutionUnit.{i}.*	Dynamic package specific parameters, applicable only to services (e.g.: status, disk space in use).

2.3 Local Management

LCM provides two local APIs. A northbound API used for interfacing with remote management clients or other services, and a southbound for internal communication with the Execution Environments (EEs).

2.3.1 Northbound API (IPC Bus)

The northbound API, exposed on the local IPC bus, enables services (e.g.: remote management protocol or local Web-GUI) to perform CRUD based operations on both Execution Environments and Packages, which resemble services.

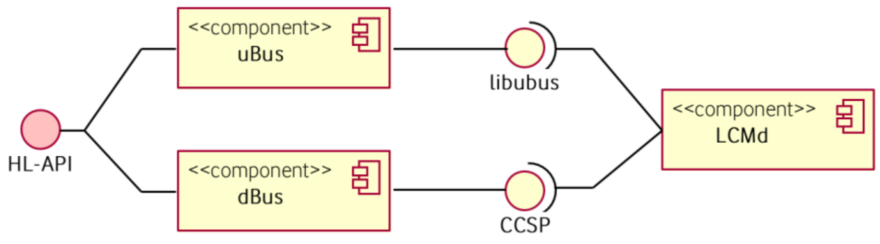
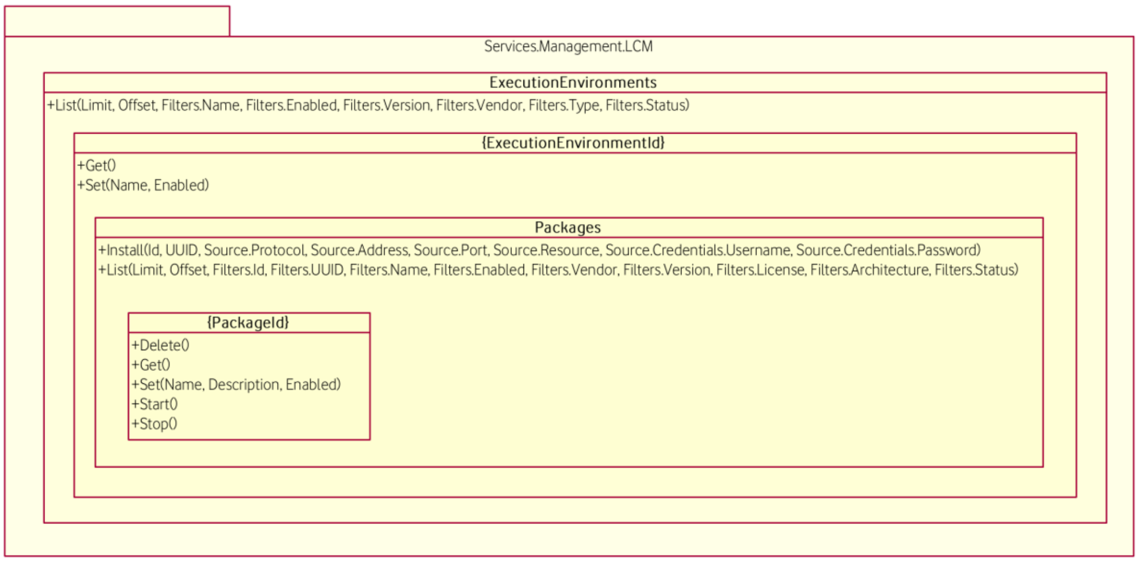


Figure 6. Northbound API Interface.

The following diagram provides an overview on the availables objects and methods.



Services.Management.LCM
ExecutionEnvironments
+List(Limit, Offset, Filters.Name, Filters.Enabled, Filters.Version, Filters.Vendor, Filters.Type, Filters.Status)
{ExecutionEnvironmentId}
+Get()
+Set(Name, Enabled)
Packages
+Install(Id, UUID, Source.Protocol, Source.Address, Source.Port, Source.Resource, Source.Credentials.Username, Source.Credentials.Password)
+List(Limit, Offset, Filters.Id, Filters.UUID, Filters.Name, Filters.Enabled, Filters.Vendor, Filters.Version, Filters.License, Filters.Architecture, Filters.Status)
{PackageId}
+Delete()
+Get()
+Set(Name, Description, Enabled)
+Start()
+Stop()

Figure 7. LCM Northbound API Methods.

2.3.1.1 ExecutionEnvironments

2.3.1.1.1 List

Retrieves a list of LCM Execution Environments.

Table 3. EE List Method Sample Request & Response.

Request Body	Response Body
{ "Limit": 10, "Offset": 0, "Filters": { "Name": "BaseSystem", "Enabled": true, "Version": "1.0", "Vendor": "prplFoundation", "Type": "LXC", "Status": "Active" } }	{ "Header": { "Name": "OK" }, "Body": { "List": [{ "Id": "0", "Name": "BaseSystem", "Enabled": true, "Version": "1.0", "Vendor": "prplFoundation", "Type": "LXC", "Resources": { "Memory": { "Total": 64000, "Free": 32000, "Usage": 0.70 }, "Storage": { "Total": 256000000, "Free": 128000000, "Usage": 0.50 } }, "Status": "Active" }], "Limit": 10, "Offset": 0 } }

2.3.1.2 ExecutionEnvironments.{ExecutionEnvironmentId}

2.3.1.2.1 Get

Retrieves the status and configuration parameters in regards to the (specified) LCM Execution Environment.

Table 4. EE Get Method Sample Request & Response.

Request Body	Response Body
{}	{ "Header": { "Name": "OK" }, "Body": { "Id": "0", "Name": "BaseSystem", "Enabled": true, "Version": "1.0", "Vendor": "prplFoundation", "Type": "LXC", "Resources": { "Memory": { "Total": 64000, "Free": 32000, "Usage": 0.70 }, "Storage": { "Total": 256000000, "Free": 128000000, "Usage": 0.50 } }, "Status": "Active" } }

2.3.1.2.2 Set

Retrieves the status and configuration parameters in regards to the (specified) LCM Execution Environment.

Table 5. EE Set Method Sample Request & Response.

Request Body	Response Body

{
  "Name": "BaseSystem",
  "Enabled": true
}
	{
	  "Header": {
	    "Name": "OK"
	  }
	}

This method execution may also lead to the following events being raised.

Table 6. EE Set Method Events.

Step	Event

1	{
	  "Header": {
	    "Code": 3,
	    "Name": "SERVICES_MANAGEMENT_LCM_EXECUTION_ENVIRONMENT_MODIFIED"
	  },
	  "Body": {
	    "ExecutionEnvironmentId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}"
	  }
	}

2a	{
	  "Header": {
	    "Code": 4,
	    "Name": "SERVICES_MANAGEMENT_LCM_EXECUTION_ENVIRONMENT_ENABLED"
	  },
	  "Body": {
	    "ExecutionEnvironmentId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}"
	  }
	}

2b	{
	  "Header": {
	    "Code": 5,
	    "Name": "SERVICES_MANAGEMENT_LCM_EXECUTION_ENVIRONMENT_DISABLED"
	  },
	  "Body": {
	    "ExecutionEnvironmentId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}"
	  }
	}

2.3.1.3 ExecutionEnvironments.{ExecutionEnvironmentId}.Packages

2.3.1.3.1 Install

Installs a new LCM Package.

Table 7. Package Install Method Sample Request & Response.

Request Body	Response Body
{ "Id": "uci", "UUID": "tOhQEAzEzk9zbf9uljt5OnjKEifP8JvQ", "Source": { "Protocol": "HTTPS", "Address": "feeds.prpl.org", "Port": "8080", "Resource": "uci.ipkg", "Credentials": { "Username": "prpl", "Password": "foundation" } } }	{ "Header": { "Name": "OK" }, "Body": { "Id": "uci" } }

This method execution may also lead to the following events being raised.

Table 8. Package Install Method Events.

Step	Event
1a	{ "Header": { "Code": 6, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DOWNLOAD_COMPLETE" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }
1b	{ "Header": { "Code": 7, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DOWNLOAD_FAILED", "Reason": "UNREACHABLE" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }

2a	{ "Header": { "Code": 8, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_INSTALL_COMPLETE" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }
2b	{ "Header": { "Code": 9, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_INSTALL_FAILED", "Reason": "DEPENDENCIES_MISSING" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }
3	{ "Header": { "Code": 1, "Name": "SERVICES_MANAGEMENT_LCM_EXECUTION_ENVIRONMENT_ADDED" }, "Body": { "ExecutionEnvironmentId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}" } }

2.3.1.3.2 List

Retrieves a list of LCM Packages.

Table 9. Package List Method Sample Request & Response.

Request Body

```
{
  "Limit": 10,
  "Offset": 0,
  "Filters": {
    "Id": "e8640310-7164-4b67-87cf-
4ba717d0f094",
    "UUID": "tOhQEAzEzk9zbf9uljt5OnjKEifP8JvQ",
    "Name": "libuci",
    "Enabled": true,
    "Vendor": "Felix Fietkau",
    "Version": "2016-07-04.1-1",
    "License": "LGPL-2.1",
    "Architecture": "brcm63xx",
    "Status": "Installed"
  }
}
```

Response Body

```
{
  "Header": {
    "Name": "OK"
  },
  "Body": {
    "List": [
      {
        "Id": "e8640310-7164-4b67-87cf-4ba717d0f094",
        "UUID": "tOhQEAzEzk9zbf9uljt5OnjKEifP8JvQ",
        "Name": "libuci",
        "Description": "C library for the Unified Configuration
Interface (UCI)",
        "Enabled": true,
        "Source": {
          "Protocol": "HTTPS",
          "Address": "feeds.prpl.org",
          "Port": "8080",
          "Resource": "uci.ipkg"
        },
        "Section": "libs",
        "Vendor": "Felix Fietkau",
        "Version": "2016-07-04.1-1",
        "Dependencies": [
          "libc",
          "libssp",
          "libubox"
        ],
        "License": "LGPL-2.1",
        "Architecture": "brcm63xx",
        "Status": "Installed",
        "Install": {
          "Timestamp": "2018-04-09T20:45:00+01:00",
          "Size": 16760
        }
      }
    ],
    "Limit": 10,
    "Offset": 0
  }
}
```

2.3.1.4 ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}

2.3.1.4.1 Delete

Deletes the specified LCM Package.

Table 10. Package Delete Method Sample Request & Response.

Request Body	Response Body

{}	{
	  "Header": {
	    "Name": "OK"
	  }
	}

This method execution may also lead to the following events being raised.

Table 11. Package Delete Method Events.

Step	Event

1a	{
	  "Header": {
	    "Code": 13,
	    "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DELETE_COMPLETE"
	  },
	  "Body": {
	    "PackageId":
	"Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	  }
	}

1b	{
	  "Header": {
	    "Code": 14,
	    "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DELETE_FAILED"
	  },
	  "Body": {
	    "PackageId":
	"Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	  }
	}

2	{
	  "Header": {
	    "Code": 2,
	    "Name": "SERVICES_MANAGEMENT_LCM_EXECUTION_ENVIRONMENT_DELETED"
	  },
	  "Body": {
	    "ExecutionEnvironmentId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}"
	  }
	}

2.3.1.4.2 Get

Retrieves the status and configuration parameters in regards to the (specified) LCM Package.

Table 12. Package Get Method Sample Request & Response.

Request Body	Response Body

{}	{
	 "Header": {
	  "Name": "OK"
	 },
	 "Body": {
	  "Id": "e8640310-7164-4b67-87cf-4ba717d0f094",
	  "UUID": "tOhQEAzEzk9zbf9uljt5OnjKEifP8JvQ",
	  "Name": "libuci",
	  "Description": "C library for the Unified Configuration Interface (UCI)",
	  "Enabled": true,
	  "Source": {
	   "Protocol": "HTTPS",
	   "Address": "feeds.prpl.org",
	   "Port": "8080",
	   "Resource": "uci.ipkg"
	  },
	  "Section": "libs",
	  "Vendor": "Felix Fietkau",
	  "Version": "2016-07-04.1-1",
	  "Dependencies": [
	   "libc",
	   "libssp",
	   "libubox"
	  ],
	  "License": "LGPL-2.1",
	  "Architecture": "brcm63xx",
	  "Status": "Installed",
	  "Install": {
	   "Timestamp": "2018-04-09T20:45:00+01:00",
	   "Size": 16760
	  }
	 }
	}

2.3.1.4.3 Set

Modifies the status and configuration parameters of the (specified) LCM Package.

Table 13. Package Set Method Sample Request & Response.

Request Body	Response Body

{
  "Name": "libuci",
  "Description": "C library for the Unified Configuration Interface (UCI)",
  "Enabled": true
}
	{
	  "Header": {
	    "Name": "OK"
	  }
	}

This method execution may also lead to the following events being raised.

Table 14. Package Set Methods Events.

Step	Event

1	{
	  "Header": {
	    "Code": 12,
	    "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_MODIFIED"
	  },
	  "Body": {
	    "PackageId":
	"Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	  }
	}

2a	{
	  "Header": {
	    "Code": 10,
	    "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_ENABLED"
	  },
	  "Body": {
	    "PackageId":
	"Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	  }
	}

2b	{
	  "Header": {
	    "Code": 11,
	    "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DISABLED"
	  },
	  "Body": {
	    "PackageId":
	"Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	  }
	}

2.3.1.4.4 Start

Starts the specified LCM Package.

Table 15. Package Start Method Sample Request & Response.

Request Body	Response Body
{}	{ "Header": { "Name": "OK" } }

This method execution may also lead to the following events being raised.

Table 16. Package Start Method Events.

Step	Event
1a	{ "Header": { "Code": 10, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_ENABLED" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }
1b	{ "Header": { "Code": 11, "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DISABLED" }, "Body": { "PackageId": "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}" } }

2.3.1.4.5 Stop

Modifies the status and configuration parameters of the (specified) LCM Package.

Table 17. Package Stop Method Sample Request & Response.

Request Body	Response Body

{}	{
	 "Header": {
	 "Name": "OK"
	 }
	 }

This method execution may also lead to the following events being raised.

Table 18. Package Stop Method Events.

Step	Event

1	{
	 "Header": {
	 "Code": 11,
	 "Name": "SERVICES_MANAGEMENT_LCM_PACKAGE_DISABLED"
	 },
	 "Body": {
	 "PackageId":
	 "Services.Management.LCM.ExecutionEnvironments.{ExecutionEnvironmentId}.Packages.{PackageId}"
	 }
	 }

2.3.2 Southbound API (stdio)

Execution Environments may expose their functionality directly on the native OS's IPC bus using the HL-API, or instead rely on LCMd to proxy and expose their functionality through a custom API (e.g.: stdio JSON), as described on this chapter.

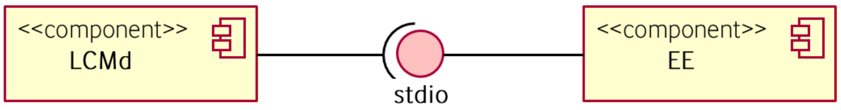
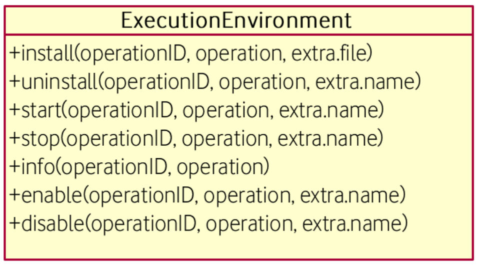


Figure 8. Southbound API Interface.

The following diagram provides an overview on the available methods.



ExecutionEnvironment

+install(operationID, operation, extra.file)
+uninstall(operationID, operation, extra.name)
+start(operationID, operation, extra.name)
+stop(operationID, operation, extra.name)
+info(operationID, operation)
+enable(operationID, operation, extra.name)
+disable(operationID, operation, extra.name)

Figure 9. Southbound API Methods.

2.3.2.1 install

Signals the execution environment to install a new package.

Table 19. Install Method Sample Request.

Request Body

{
  "operationID": "asd290fkl2kljsdf",
  "operation": "**install**",
  "extra": {
    "file": "/tmp/package.ipk"
  }
}

Table 20. Install Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “install”.	String	No
extra.file	Path of the package to be installed.	String	No

This method execution may also lead to the following events being raised.

Table 21. Install Method Sample Events.

Step	Event

1	{
	  "operationID": "asd290fkl2kljsdf",
	  "status": "**operation_started**"
	}

2a	{
	  "operationID": "asd290fkl2kljsdf",
	  "status": "**operation_completed**",
	  "extra": {
	    "name": "test_package",
	    "vendor": "stetel",
	    "version": "1.0",
	    "installed_size": "101284",
	    "description": "Test application",
	    "architecture": "armV7",
	    "depends": "none",
	    "license": "bsd",
	    "section": "utils",
	    "source": "none"
	  }
	}

2b	{
	  "operationID": "asd290fkl2kljsdf",
	  "status": "**operation_failed**",
	  "error_message": "No space left on the device"
	}

Table 22. Install Method Event Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package installing status.	Enum
extra.name	Package name.	String
extra.vendor	Package vendor.	String
extra.version	Package version.	String
extra.installed_size	Package installation size.	String
extra.description	Package description.	String
extra.architecture	Package architecture.	String
extra.depends	Package list of dependencies.	String
extra.license	Package license.	String
extra.section	Package section.	String
extra.source	Package source.	String
error_message	Description of the fault that occurred during the installation process.	String

2.3.2.2 uninstall

Signals the execution environment to uninstall and delete an existing package.

Table 23. Uninstall Method Sample Request.

Request Body

{
"operationID": "asd290fkl2kljsdf",
"operation": "**uninstall**",
"extra": {
"name": "package_name"
}
}

Table 24. Uninstall Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “uninstall”.	String	No
extra.name	Name of the package to be uninstalled.	String	No

This method execution may also lead to the following events being raised.

Table 25. Install Method Sample Events.

Step	Event

1	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_started**"
	}

2a	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_completed**"
	}

2b	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_failed**",
	"error_message": "Unable to uninstall, process locked"
	}

Table 26. Uninstall Method Event Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package uninstall status.	Enum
error_message	Description of the fault that occurred during the uninstall process.	String

2.3.2.3 start

Signals the execution environment to start a package or service.

Table 27. Start Method Sample Request.

Request Body

{
"operationID": "asd290fkl2kljsdf",
"operation": "**start**",
"extra": {
"name": "package_name"
}
}

Table 28. Start Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to "uninstall".	String	No
extra.name	Name of the package to be initialized.	String	No

This method execution may also lead to the following events being raised.

Table 29. Start Method Sample Events.

Step	Event

1	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_started**"
	}

2a	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_completed**"
	}

2b	{
	"operationID": "asd290fkl2kljsdf",
	"status": "**operation_failed**",
	"error_message": "Unable to start service"
	}

Table 30. Start Method Event Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package start status.	Enum
error_message	Description of the fault that occurred during the initialization process.	String

2.3.2.4 stop

Signals the execution environment to stop a package or service.

Table 31. Stop Method Sample Request.

Request Body
{ "operationID": "asd290fkl2kljsdf", "operation": "**stop**", "extra": { "name": "package_name" } }

Table 32. Stop Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “stop”.	String	No
extra.name	Name of the package to be initialized.	String	No

This method execution may also lead to the following events being raised.

Table 33. Stop Method Sample Events.

Step	Event
1	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_started**" }
2a	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_completed**" }
2b	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_failed**", "error_message": "Unable to stop service" }

Table 34. Stop Method Response Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package stop status.	Enum
error_message	Description of the fault that occurred during the stop process.	String

2.3.2.5 info

Signals the execution environment to retrieve a package configuration details.

Table 35. Info Method Request Sample.

Request Body

{
 "operationID": "asd290fkl2kljsdf",
 "operation": "**info**"
}

Table 36. Info Method Response Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “info”.	String	No

Table 37. Stop Method Sample Events.

Step	Event

1	{
	 "operationID": "asd290fkl2kljsdf",
	 "status": "**operation_started**"
	}

2	{
	 "operationID": "asd290fkl2kljsdf",
	 "status": "**operation_completed**",
	 "extra": {
	  "Name": "Supervisor",
	  "Version": "1.0",
	  "Vendor": "VendorName",
	  "Type": "VendorName Supervisor v1.0",
	  "Memory": {
	   "Total": "52428",
	   "Free": "51036",
	   "Usage": "0.026562"
	  },
	  "Storage": {
	   "Total": "7817134",
	   "Free": "6176124",
	   "Usage": "0.209925"
	  }
	 }
	}

Table 38. Info Method Event Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package stop status.	Enum
extra.Name	Package name.	String
extra.Version	Package version.	String
extra.Vendor	Package vendor.	String
extra.Type	Package type.	String
extra.Memory.Total	Total available volatile memory in bits.	Integer
extra.Memory.Free	Free available volatile memory in bits.	Integer
extra.Memory.Usage	Volatile memory usage ratio.	Float
extra.Storage.Total	Total available persistent storage in bits.	Integer
extra.Storage.Free	Free available persistent storage in bits.	Integer
extra.Storage.Usage	Persistent storage usage ratio.	Float

2.3.2.6 enable

Signals the execution environment to enable a package or service.

Table 39. Enable Method Sample Request.

Request Body
{ "operationID": "asd290fkl2kljsdf", "operation": "**enable**", "extra": { "name": "package_name" } }

Table 40. Enable Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “enable”.	String	No
extra.name	Name of the package to be initialized.	String	No

This method execution may also lead to the following events being raised.

Table 41. Enable Method Sample Events.

Step	Event
1	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_started**" }
2a	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_completed**" }
2b	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_failed**", "error_message": "Unable to enable service" }

Table 42. Enable Method Response Body Parameters.

Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package stop status.	Enum
error_message	Description of the fault that occurred during the stop process.	String

2.3.2.7 disable

Signals the execution environment to disable a package or service.

Table 43. Disable Method Sample Request.

Request Body
{ "operationID": "asd290fkl2kljsdf", "operation": "**disable**", "extra": { "name": "package_name" } }

Table 44. Disable Method Request Body Parameters.

Parameter	Description	Type	Optional
operationID	Unique operation identified used for matching responses.	String	No
operation	Operation to be triggered. Value must be set to “disable”.	String	No
extra.name	Name of the package to be initialized.	String	No

This method execution may also lead to the following events being raised.

Table 45. Disable Method Sample Events.

Step	Event
1	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_started**" }
2a	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_completed**" }
2b	{ "operationID": "asd290fkl2kljsdf", "status": "**operation_failed**", "error_message": "Unable to disable service" }

Table 46. Disable Method Response Body Parameters.

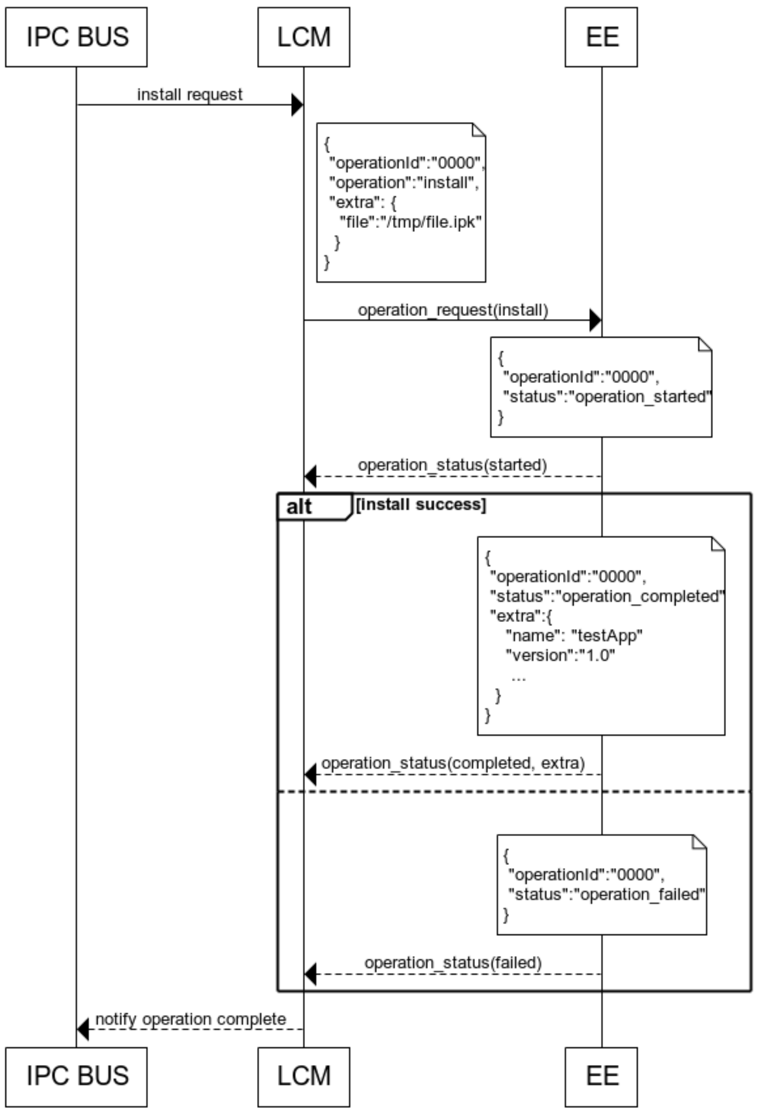
Parameter	Description	Type
operationID	Unique operation identified used for matching responses.	String
status	Package stop status.	Enum
error_message	Description of the fault that occurred during the stop process.	String

3 Appendix

3.1 Sequence Flows

3.1.1 Execution Environment

3.1.1.1 Install


IPC BUS
LCM
EE
install request
operation_request(install)
operation_status(started)
alt
[install success]
operation_status(completed, extra)
operation_status(failed)
notify operation complete
IPC BUS
LCM
EE

```
{
 "operationId":"0000",
 "operation":"install",
 "extra": {
   "file":"/tmp/file.ipk"
  }
}
```

```
{
 "operationId":"0000",
 "status":"operation_started"
}
```

```
{
 "operationId":"0000",
 "status":"operation_completed"
 "extra":{
    "name": "testApp"
    "version":"1.0"
     ...
   }
}
```

```
{
 "operationId":"0000",
 "status":"operation_failed"
}
```

Figure 10. Execution Environment Install Method Sequence Diagram.

3.1.1.2 List

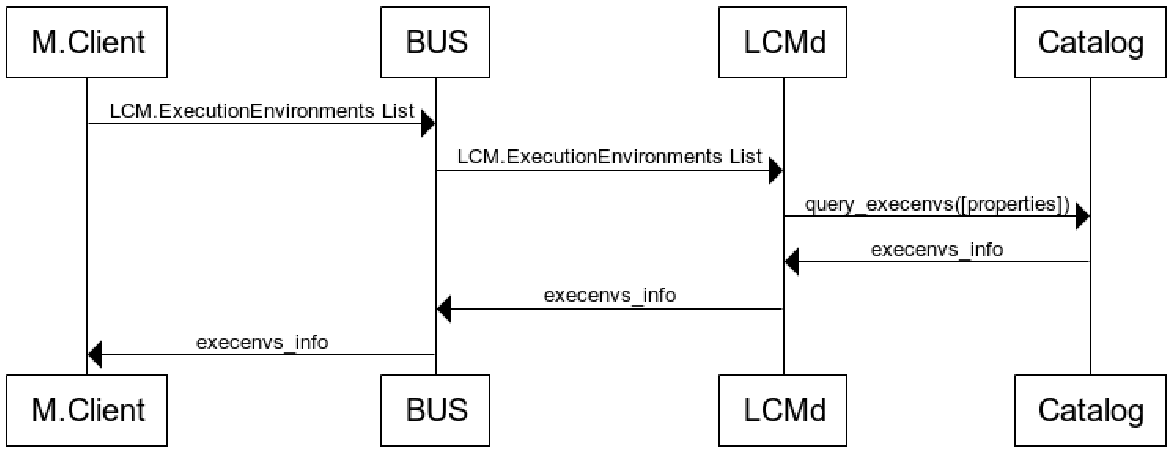

M.Client
BUS
LCMd
Catalog
LCM.ExecutionEnvironments List
LCM.ExecutionEnvironments List
query_execenvs([properties])
execenvs_info
execenvs_info
execenvs_info
M.Client
BUS
LCMd
Catalog

Figure 11. Execution Environment List Method Sequence Diagram.

3.1.2 Packages

3.1.2.1 List

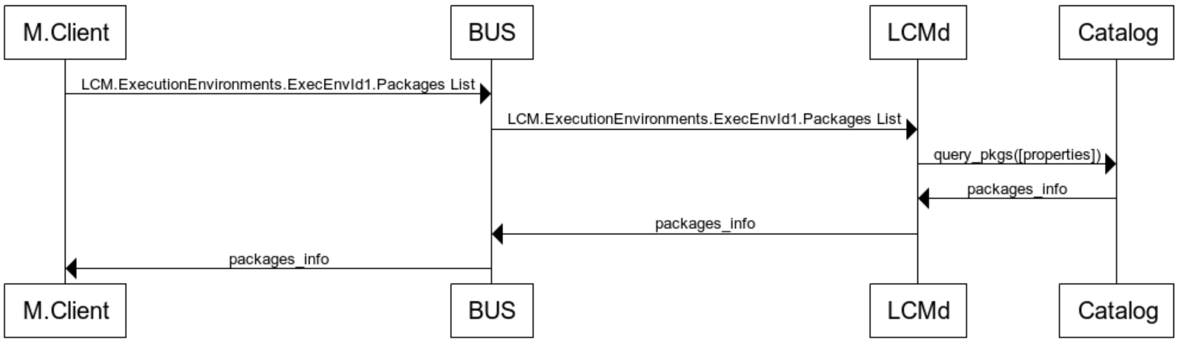


Figure 12. Package List Method Sequence Diagram.

3.1.2.2 Modify

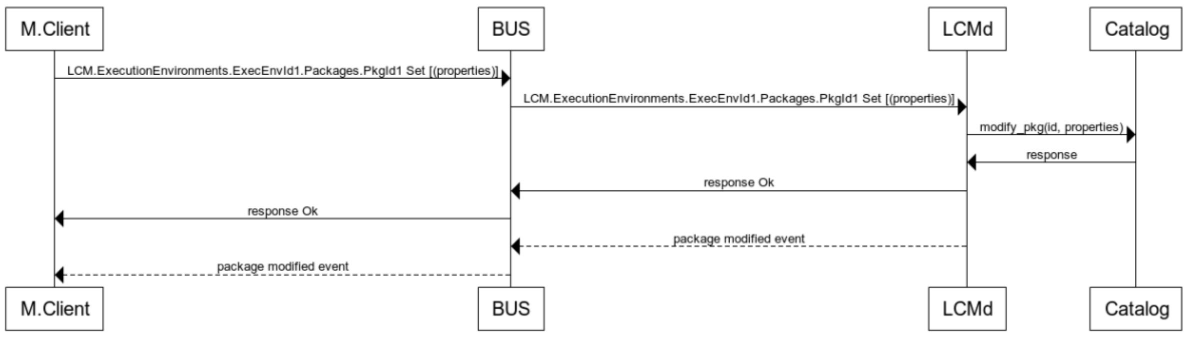

M.Client
BUS
LCMd
Catalog
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Set [(properties)]
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Set [(properties)]
modify_pkg(id, properties)
response
response Ok
response Ok
package modified event
package modified event
M.Client
BUS
LCMd
Catalog

Figure 13. Package Modify Method Sequence Diagram.

3.1.2.3 Install

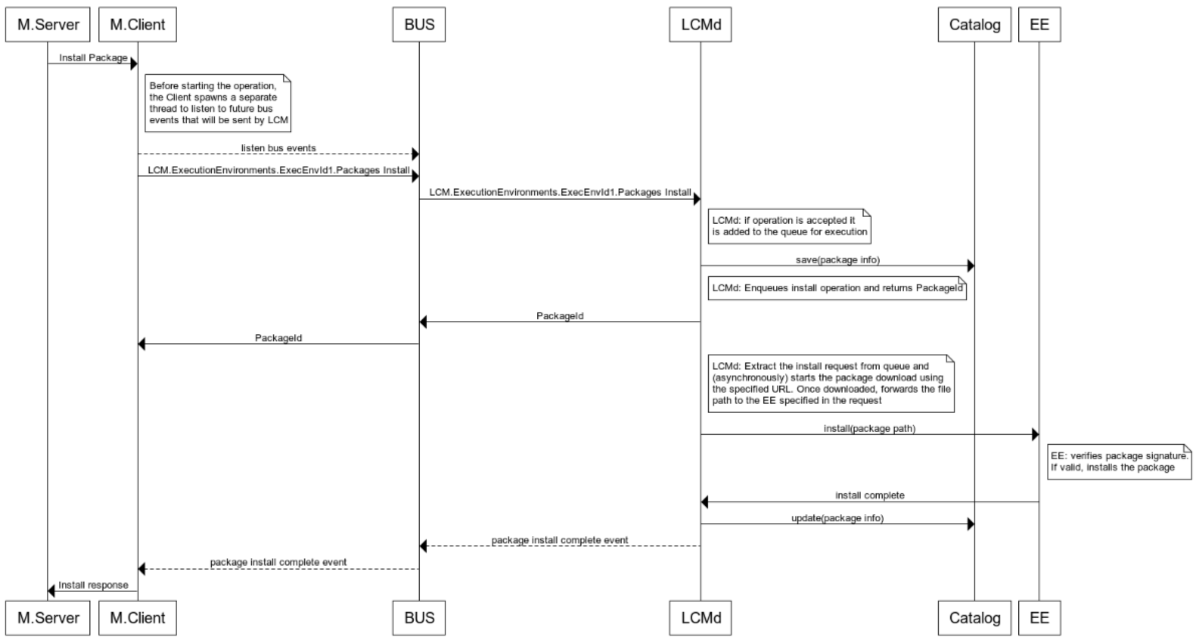

M.Server
M.Client
BUS
LCMd
Catalog
EE
Install Package
Before starting the operation, the Client spawns a separate thread to listen to future bus events that will be sent by LCM
listen bus events
LCM.ExecutionEnvironments.ExecEnvId1.Packages Install
LCM.ExecutionEnvironments.ExecEnvId1.Packages Install
LCMd: if operation is accepted it is added to the queue for execution
save(package info)
LCMd: Enqueues install operation and returns PackageId
PackageId
PackageId
LCMd: Extract the install request from queue and (asynchronously) starts the package download using the specified URL. Once downloaded, forwards the file path to the EE specified in the request
install(package path)
EE: verifies package signature. If valid, installs the package
install complete
update(package info)
package install complete event
package install complete event
Install response
M.Server
M.Client
BUS
LCMd
Catalog
EE

Figure 14. Package Install Method Sequence Diagram.

3.1.2.4 Uninstall

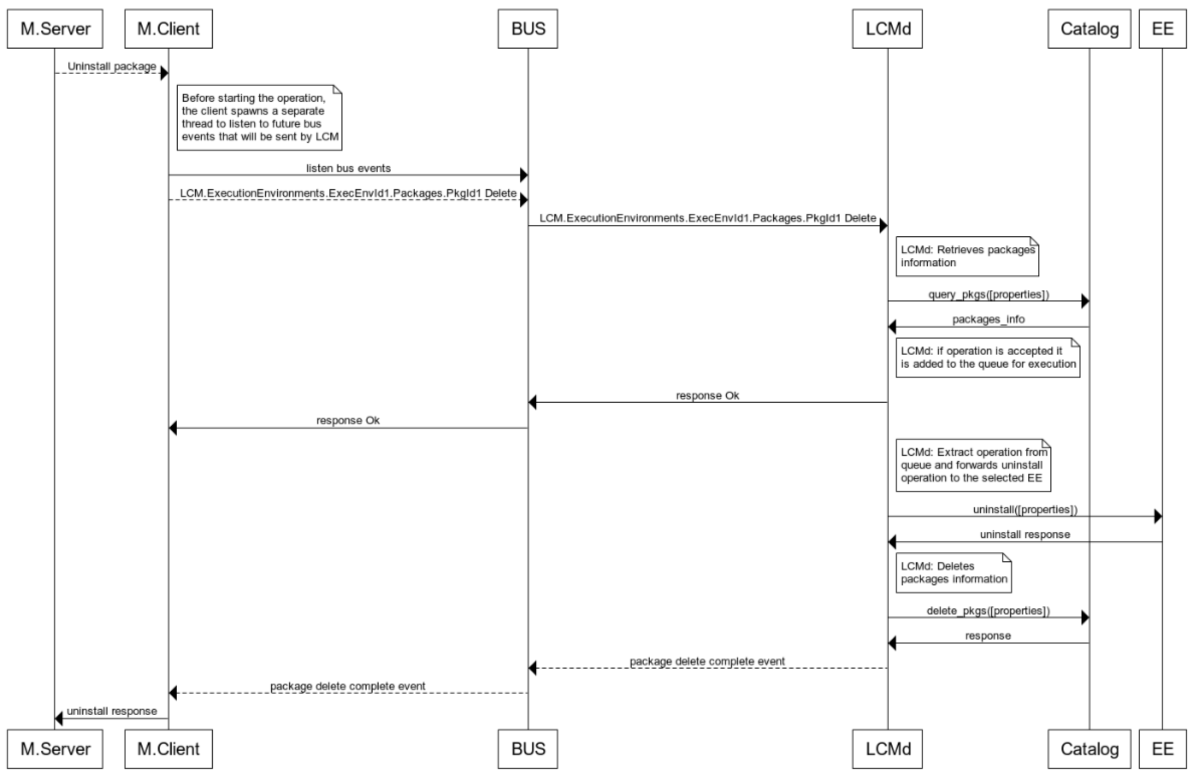

M.Server
M.Client
BUS
LCMd
Catalog
EE
Uninstall package
Before starting the operation, the client spawns a separate thread to listen to future bus events that will be sent by LCM
listen bus events
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Delete
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Delete
LCMd: Retrieves packages information
query_pkgs([properties])
packages_info
LCMd: if operation is accepted it is added to the queue for execution
response Ok
response Ok
LCMd: Extract operation from queue and forwards uninstall operation to the selected EE
uninstall([properties])
uninstall response
LCMd: Deletes packages information
delete_pkgs([properties])
response
package delete complete event
package delete complete event
uninstall response

Figure 15. Package Uninstall Method Sequence Diagram.

3.1.2.5 Start

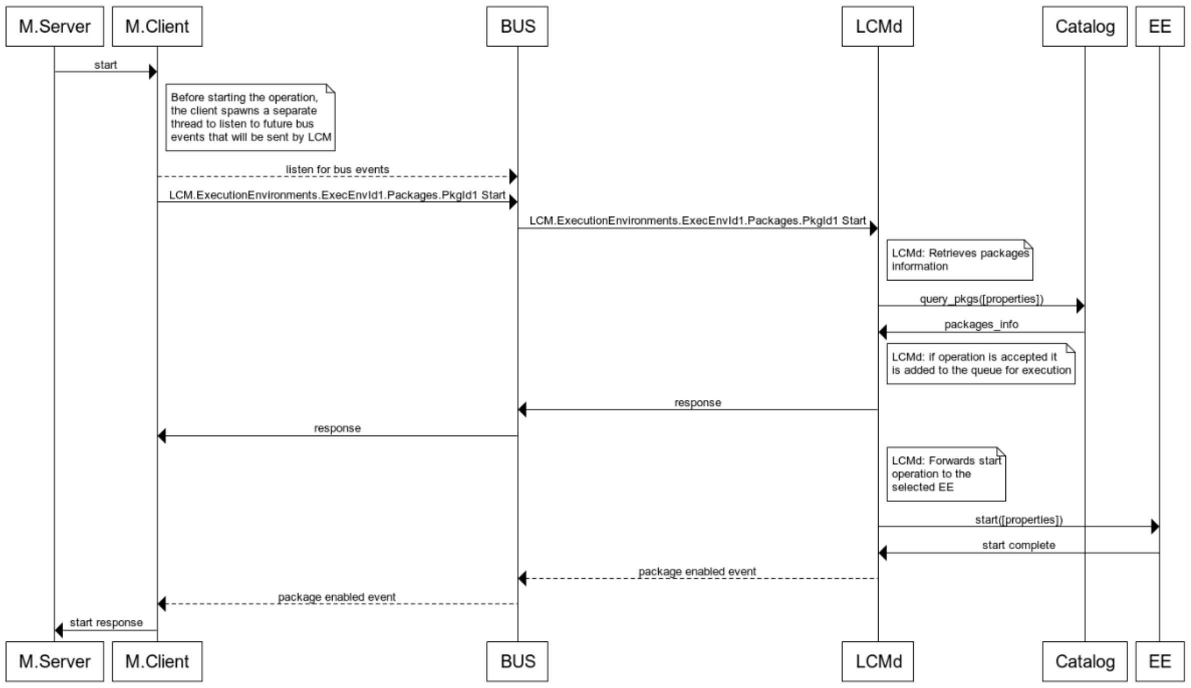

M.Server
M.Client
BUS
LCMd
Catalog
EE
start
Before starting the operation, the client spawns a separate thread to listen to future bus events that will be sent by LCM
listen for bus events
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Start
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Start
LCMd: Retrieves packages information
query_pkgs([properties])
packages_info
LCMd: if operation is accepted it is added to the queue for execution
response
response
LCMd: Forwards start operation to the selected EE
start([properties])
start complete
package enabled event
package enabled event
start response
M.Server
M.Client
BUS
LCMd
Catalog
EE

Figure 16. Package Start Method Sequence Diagram.

3.1.2.6 Stop

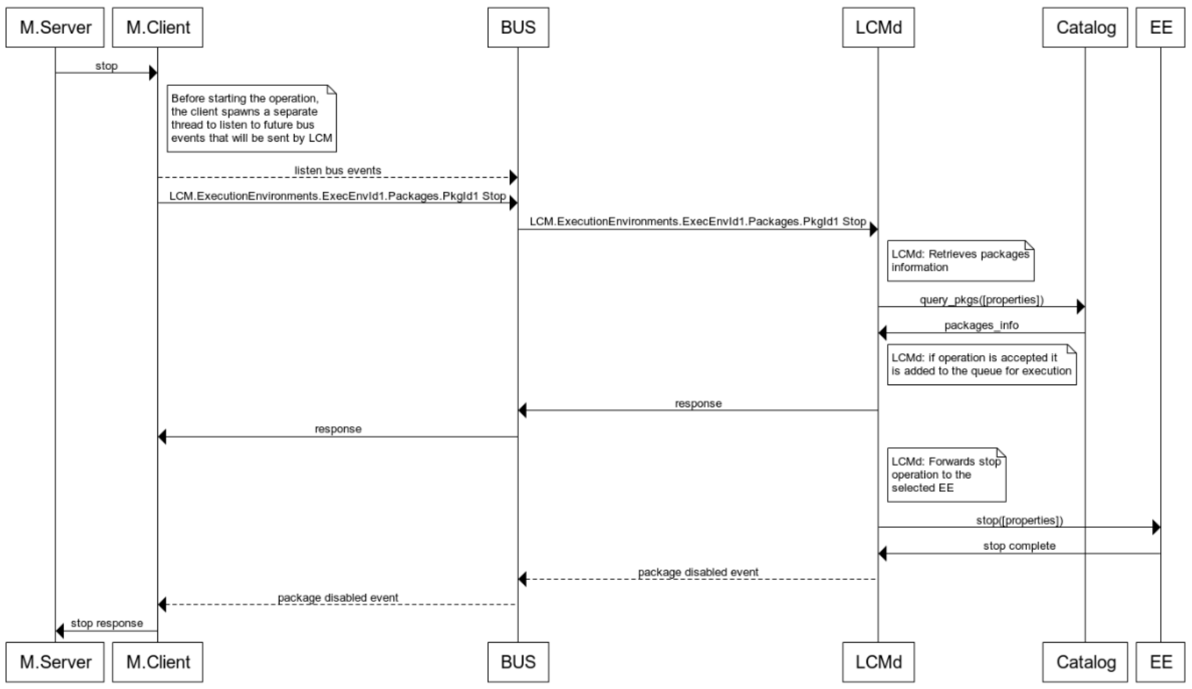

M.Server
M.Client
BUS
LCMd
Catalog
EE
stop
Before starting the operation, the client spawns a separate thread to listen to future bus events that will be sent by LCM
listen bus events
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Stop
LCM.ExecutionEnvironments.ExecEnvId1.Packages.PkgId1 Stop
LCMd: Retrieves packages information
query_pkgs([properties])
packages_info
LCMd: if operation is accepted it is added to the queue for execution
response
response
LCMd: Forwards stop operation to the selected EE
stop([properties])
stop complete
package disabled event
package disabled event
stop response
M.Server
M.Client
BUS
LCMd
Catalog
EE

Figure 17. Package Stop Method Sequence Diagram.

3.2 State Machine Diagram

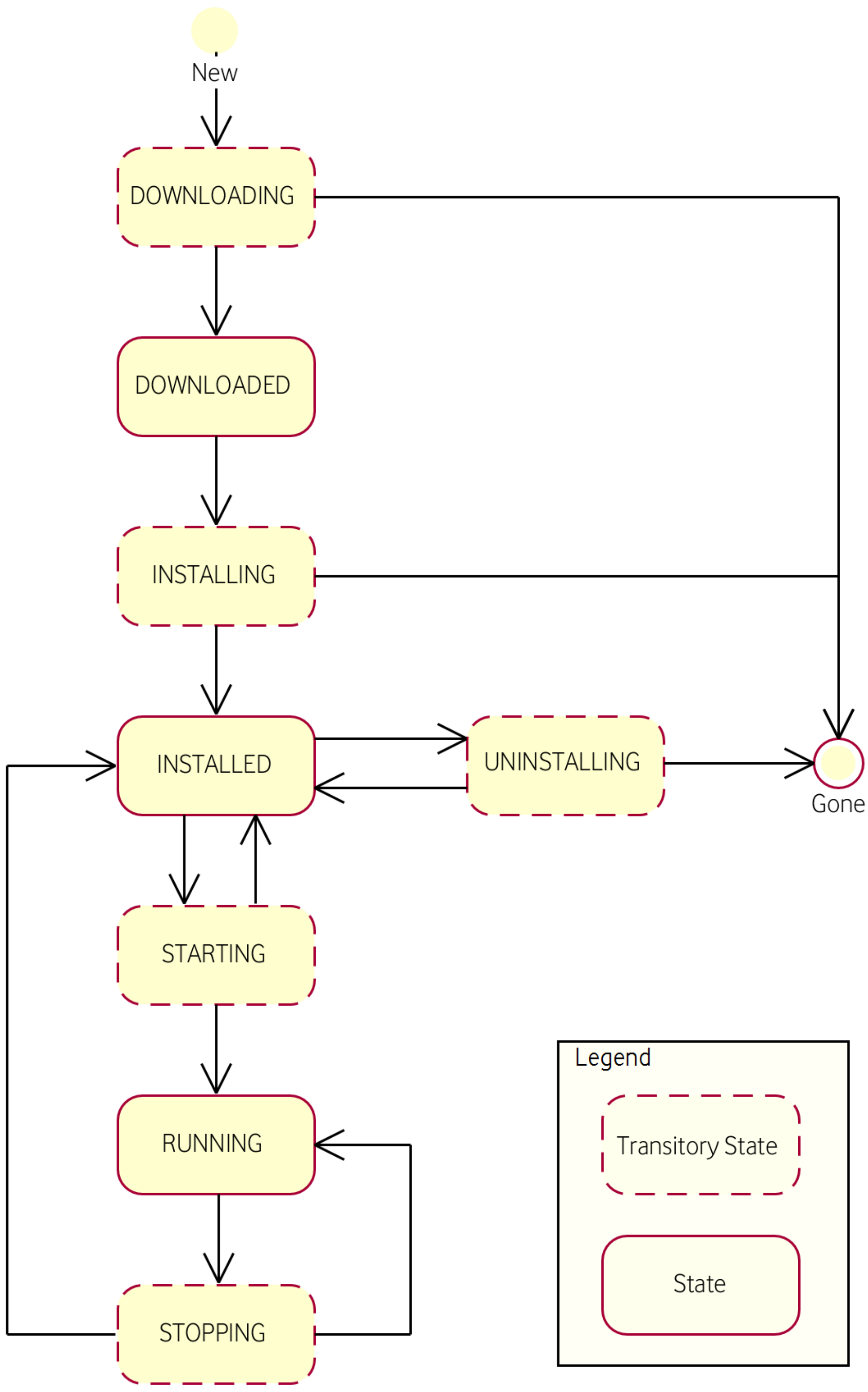


Figure 18. Package State Machine Diagram.